#11

BRIEF ON APPEAL

William C. Roch
Attorney for Appellant
Registration No. 24,972

SCULLY SCOTT MURPHY & PRESSER
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343

TABLE OF CONTENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Henry E. Butterworth, et al.	**Examiner:**	Jungwon Chang
Serial No:	09/551,962	**Art Unit:**	2154
Filed:	April 19, 2000	**Docket:**	GB919990117US1 (16891)
For:	DATA PROCESSING SYSTEM WITH MASTER AND SLAVE PROCESSORS	**Dated:**	

Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

BRIEF ON APPEAL

Sir:

This is a BRIEF ON APPEAL in support of applicant's appeal from the Examiner's decision dated November 20, 2003 finally rejecting claims 1-4. This BRIEF ON APPEAL is arranged in compliance with 37 C.F.R. §1.192(c), with subheadings and the numbers of the subheadings in conformance therewith.

Claims 1-4 are set forth in APPENDIX A attached hereto, as specified by 37 C.F.R. §1.192(c)(9). The explanation in the Final Rejection of November 20, 2003 of the rejection of claims 1-4 under 35 U.S.C. §103 is attached hereto in APPENDIX B.

CERTIFICATE OF MAILING UNDER 37 C.F.R. §1.8(a)

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner of Patents, P. O. Box 1450, Alexandria, VA 22313-1450 on 4/15, 2004.

Dated: 4/15, 2004

William C. Roch
William C. Roch

G:\Ibm\105\16891\Amend\16891.appealbrief.doc

1. REAL PARTY IN INTEREST

The real party in interest is the assignee International Business Machines Corporation.

2. RELATED APPEALS AND INTERFERENCES

No related appeals or interferences are known to the appellant, or the appellant's legal representatives, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

3. STATUS OF CLAIMS

Claims 1-4 are pending in this patent application, and are involved in this appeal.

4. STATUS OF AMENDMENTS

No AMENDMENTS have been filed subsequent to the Final Rejection. A RESPONSE UNDER 37 CFR 1.116 was filed on January 20, 2004 that requested reconsideration of the prior art rejection in the Final Rejection. An Advisory Action of February 18, 2004 advised that the request was nonpersuasive.

5. SUMMARY OF THE INVENTION

The problem that the present invention seeks to solve is to allow a master high performance processor to access a high latency memory without stalling the processor. The solution is to use a slave processor to access the memory, freeing the master processor to continue other processing operations and tasks while the slave processor accesses the memory and until the slave processor later posts the results of the reads to the master processor. p (page) 11, l (lines) 8-24.

There has been a rapid increase in the clock speed of microprocessors. Presently available examples, such as the PowerPC 750 processor available from International Business Machines Corporation, can operate at clock speeds of 400MHz or more. As the speed of microprocessors has increased, the price of such microprocessors has decreased. It is now practical to embed high performance microprocessors in controller systems such as disk controller systems. p 1, l 10-18.

Such controller systems typically comprise a microprocessor or microcontroller connected to a memory subsystem and one or more controlled devices via an external bus system. The external buses usually operate at slower speeds than the processor. Therefore, it is difficult to operate all elements of the controller system at the same high frequency. As more devices are attached to the bus, the load imposed on bus drivers of the external bus system is correspondingly increased. The increased load leads to a corresponding reduction in clock speed. p 1, l 20 to p 2, l 2.

In some controller systems, the external bus system comprises a hierarchy of buses interconnected by bridge devices. The bridge devices isolate signals carried by the buses in the hierarchy. Each bus in the hierarchy interconnects a different group of devices. This arrangement reduces the effective load on the external bus system. A reasonable clock speed can thus be maintained. However, the intermediate bridge devices introduce a delay or latency to memory accesses. These latencies reduce the processing speed of the processor, particularly in connection with communications between the processor and devices connected to extremes of the bus system. Further, there is usually arbitration associated with accesses to each bus in the bus system. Bus arbitration adds further latency. Further still, some buses, such as PCI buses, operate asynchronously relative to the devices they interconnect. Such asynchronous operation leads to introduction of further latency because additional clock cycles are used in providing valid data signals on the buses. p 2, l 4-23.

When the processor issues a write command, it is possible to reduce delay introduced by the bridge devices via buffers in the bridge devices. The buffers enable the waiting or "stall" time in the processor to be reduced to the time taken to send the write command through the bus which is directly connected to the processor. For example, caches in the memory subsystem usually send or "post" data cache line flush commands to the memory subsystem thereby enabling the processor to continue operation even as data is transferred between successive layers of the memory subsystem. Such "posted write" systems are well known. p 2, l 25 to p 3, l 8.

For read commands however, there is no such satisfactory solution. In general, a read command incurs all the delays introduced by arbitration and synchronization at each bridge device in the bus system. Also, prior to execution of a read command, all posted write commands within each bridge device should be completed in case the region of memory being read is affected. p 3, l 10-17.

In the PCI-X bus system, strict ordering of read and posted write commands can be relaxed to reduce latency in the execution of read commands. However, this can adversely affect execution of the program code by the processor. Also, the latency inherent in read commands is still substantial. p 3, l 19-24.

Further difficulties arise if accesses through the bridge devices become too frequent and begin to approach the bandwidth of the intervening buses in the bus system. This situation arises when, for example, many short random accesses are made to remote memory locations. This problem can be alleviated by providing sufficient memory as close as possible to the processor. For example, the system may be provided with large Ll and L2 caches. However, in many controller systems, it is also desirable to place at least .some memory close to other devices of the controller system, such as device interfaces. Such interfaces are also intolerant of latencies. Thus, if the memory is too remote from these devices, performance will be adversely affected. p 3, l 28 to p 4, l 11.

In a typical controller system, data which is written by a device interface and read by the processor is posted through bridge devices to memory located close to the processor. Memory which is written to by the processor, and read by the device interface chip is located close to the device interface chip. The amount of data that can be transferred between the processor and the device interface is limited to the capacity of the intervening bus system. Also, at least some memory must be both read from and written to by one or both of the processor and the device interface chip. Furthermore, there are practical limitations in terms of both addressability and cost on the amount of memory that can be assigned to the processor.

An embedded disk controller typically requires a cache which is too large for direct connection to a fast processor at a reasonable cost. Conventionally, therefore, such caches have been attached to the processor via a memory controller remote from the processor. A problem associated with this arrangement is that accesses to the remote memory are very slow, typically taking between hundreds and thousands of clock cycles of the processor. Such times can exceed the execution time or code waiting to be executed by the processor. The imbalance between memory access time and processor cycle time increases as microprocessors get faster. p 4, l 13 to p 5, l 10.

The present invention provides a data processing system, such as a disk controller system, comprising a master processor, a slave processor, a memory, and a bus subsystem interconnecting the master processor, the slave processor, and the memory; wherein the master processor is configured to generate, in response to a memory access instruction, a read request comprising a read command for execution by the slave processor to read data stored in a location in the memory specified by the memory access instruction, and to write the read request to the slave processor via a bus subsystem, and the slave processor is configured to execute the read command received in the read request from the master processor to obtain the data stored at the specified location in the memory and to write the data obtained to the master processor via the bus subsystem.

The present invention is particularly applicable to a mass storage controller system, such as a disk controller system. p 6, l 1-5.

The present invention provides a method for reading data from a memory in a data processing system, such as a disk controller system, wherein the master processor, in response to a memory access instruction, generates a read request comprising a read command for execution by the slave processor to read data stored in a location in the memory specified by the memory access instruction; writing, by the master processor, the read request to the slave processor via the bus subsystem; executing, by the slave processor, the read command received in the read request from the master processor to obtain the data stored at the specified location in the memory; and, writing the data obtained to the master processor via the bus subsystem. p 6, l 8-24.

A preferred embodiment of the present invention provides a controller system having a relatively high performance master processor, and a relatively low performance slave processor all interconnected by a bus subsystem. The slave processor is disposed relative to the memory such that it experiences less read latency and uses less bus bandwidth during memory accesses than the master processor. In a particularly preferred embodiment, the slave processor is embedded in a bridge device of the bus subsystem located closer to the memory. In use, the slave processor alleviates the burden of memory accesses otherwise imposed on the master processor. Thus, much of the cycle time of the slave processor is stalled on memory accesses. The slave processor therefore need not be especially sophisticated or fast. p 6, l 26 to p 7, l 13.

When the master processor requires data either to be read from or written to the memory, it issues a write command containing the required memory access to the slave processor. In receipt of such a write command, the slave processor performs the specified memory access on behalf of the master processor. Operations such as cache directory lookups can involve a large number of random memory accesses with which a high latency is typically associated. In preferred embodiments of the present invention, such a directory lookup request may be encoded in a message by the master processor and passed to the slave processor for handling. The slave processor then executes the code associated with the message to perform the lockup, absorb the latency associated with the lockup operation, and return the result of the lockup operation to the master processor. In the mean time, the master processor is free to perform other tasks. p 7, l 24 to p 8, l 13.

The slave processor can also improve system performance in performing other management tasks associated with the device interface chips. For example, the slave processor can be instructed to construct requests to device interface chips and handle data produced on execution of such requests by the device interface chips. In this arrangement, only abbreviated information need be communicated between the slave processor and the main processor. Thus traffic on the intervening bus subsystem can be reduced. p 8, l 15-24.

Figure 1 illustrates an example of a data processing system of the present invention that comprises a master processor 10 coupled to a bus subsystem 20. The bus subsystem 20 is also coupled to a memory 50 via a bridge device 40. A slave processor 30 is also connected to the bus subsystem 20. The master processor 10 may be implemented by a microprocessor such as a PowerPC processor 750 available from International Business Machines Corporation operating at a clock frequency of 400MHz. The slave processor 30 may also be implemented by a microprocessor such as a PowerPC processor 403 available from International Business Machines Corporation operating at a clock frequency of 100MHz. p 9, l 22 to p 10, l 6.

Figure 1 illustrates the slave processor 30 as being separate from the bridge device 40. However, referring to now to Figure 2, in other embodiments of the present invention, the slave processor 30 and the bridge device 40 may be integrated in a single chip package 60. In a particularly preferred embodiment of the present invention, the slave processor 30 is embedded in the bridge device 40. p 10, l 9-17.

Referring to Figure 3, when the master processor 10 requires data stored in the memory 60, it writes a read command 70 to the slave processor 40 via the bus subsystem 20. The read command 70 specifies the location in memory 50 to be read. The slave processor 30 executes the read command 70 received from the master processor 10 to retrieve, at 80, the required data, at 90, from the memory 50 via the bridge device 40 and the bus subsystem 20. The slave processor 30 then writes, at 100, the retrieved data to the master processor 10 via the bus subsystem 20. p 10, l 24 to p 11, l 6.

Referring to Figure 4, when data is required to be read from the memory 50, the master processor 20 generates, at 110, a corresponding read request for execution by the slave processor 30. At 120, the master processor 10 writes the read request to the slave processor. Advantageously, the master processor 10 does not then have to wait or otherwise stall until the requested data is returned from the memory 50. This is because the master processor 10 has effectively delegated execution of the read request to the slave processor 20. The slave processor 30 absorbs any waiting, stalling or other latency associated with the memory access. Thus, at 130, once the read request is sent, the master processor 10 is free to perform other tasks, at least until the requested data arrives from the slave processor 30. p 11, l 10-24.

Referring to Figure 5, at 150, on receipt of a read request from the master processor 10, the slave processor 30 extracts the read command from the read request to determine the location or locations of the memory 50 from which data is to be read . At 160, the slave processor 30 executes the extracted read command to obtain the required data from the memory 50 via the bus subsystem 20 and the bridge device 40. The slave processor 30 absorbs any latency or waiting associated with execution of the read command. At 170, the slave processor 30 generates a message containing the data obtained from the memory 50 and, at 180, writes the message to the master processor 10, thereby supplying the master processor 10 with the data originally requested. P 11, l 26 to p 12, l 13.

Referring to Figure 6, a mass storage system embodying the present invention comprises a controller 220 comprising an adapter card installed in the bus architecture of a host computer system 210. Examples of such bus architectures includes the well known PCI, ISA plug and play and EISA bus architectures. The controller 220 is coupled to a remote mass storage device 190 via a communication link 200 such as a SCSI or SSA communication link. The mass storage device 190 comprises one or more storage elements such as magnetic or optical disks or the like. p 12, l 18 to p 13, l 2

Figure 7, illustrates a mass storage system having a controller 270 including a data processing system installed in a remote unit 290, in which there is also housed a network adapter 250 and a mass storage device 280. Both the network adapter 250 and the mass storage device 280 are connected to the controller 270. The network adapter 250 is also connected, via a communication link 260 such as a Fiber Channel communication link to another network adapter 240 installed in a host computer 230. p 13, l 2-13.

6. CONCISE STATEMENT OF THE ISSUES PRESENTED FOR REVIEW

Whether claims 1-4 are unpatentable under 35 U.S.C. §103 as being obvious over Platko et al (U.S. 6,330,658) in view of Kozlowski et al. (U.S. 6,513,070).

7. GROUPING OF CLAIMS

Claims 1 and 2 are similar independent system claims and are believed to present similar issues of patentability. Claim 4 is an independent method claim, and contains many limitations similar to claims 1 and 2. However, claim 4 may present different issues than claims 1 and 2 in the event that some of the limitations of claims 1 and 2 are possibly construed to be functional limitations, while similar recitations in claim 4 are considered to be proper method limitations. To that extent, claim 4 may be separately patentable from claims 1 and 2, and does not stand or fall together with claims 1 and 2.

8. <u>APPELLANT'S ARGUMENTS WITH RESPECT TO EACH OF THE ISSUES ON APPEAL</u>

<u>Platko et al</u>

The cited prior art Platko interfaces a slave processor (e.g. encryption engine) to an ASIC (master processor) at a lower cost, by <u>reusing the memory bus</u> and employing some extra signals. <u>Reusing the memory bus reduces the number of pins consumed on the ASIC</u>.

However, the Platko reference does not allow the master processor to continue other processing operations and tasks while the slave processor is reading the memory; indeed, Platko requires that the master processor in one scenario actually drive the reads that generate data onto the memory bus, and then further drives the writes that put the data into the slave processor, and in another scenario use reads and writes to transfer the data from slave processor to memory.

Though the same terms are used in the present invention and Platko, the "master processor" in the present invention is very different from the "master processor" in the prior art Platko reference which is actually an ASIC with custom logic. In Platko, only a portion of this custom logic need be involved in data transfers, and it is therefore relatively inexpensive to stall operations of the ASIC in marshalling these data transfers. In contrast thereto, in the present invention the "master processor" is a high performance processor, which is entirely consumed while waiting for the data transfer, and is therefore expensive.

So, a key difference is that the master processor of the present invention is able to continue processing operations while the read operation is being performed by the slave processor.

Those features of the present invention are mentioned in the specification at page 8, lines 12-13, "In the meantime the processor is free to perform other tasks," and at page 11, lines 15-21, "Advantageously, the master processor 10 does not then have to wait or otherwise stall until the requested data is returned from the memory 50. This is because the master processor 10 has effectively delegated execution of the read request to the slave processor 20. The slave processor 30 absorbs any waiting, stalling or other latency associated with the memory access."

In the cited prior art reference Platko, the master processor is synchronously sequencing the operation of the slave processor. Whereas, in the present invention, the master processor is sending an asynchronous request to the slave processor and the slave processor is coming back with an asynchronous reply. In summary, the present invention avoids stalling the master processor while the asynchronous memory read is in progress.

Platko concerns, col. 1, lines 27-40, "a network interface card (NIC) of the type used in host systems such as personal computers and workstations. NICs are generally plug-in circuit cards having an interface to an I/O bus used in the host system, along with an interface to a physical network medium. In an NIC, it is common to employ random access memory (RAM) as a temporary buffer storage for packets that have been received from the network or that are to be transmitted on the network. Along with the buffer RAM, the NIC contains a significant amount of complex logic for implementing the respective interfaces to the host I/O bus and the network, and to move data along respective data paths between the I/O bus and the buffer RAM, and between the network and the buffer RAM."

In Platko, col. 2, lines 12-16, "an optional co-processor is supported <u>without requiring a separate interface</u> on a master processor. High system performance is achieved, while device cost and complexity are reduced by <u>keeping pin counts relatively low</u>.

In Platko, col. 2, lines 30-49, "The master processor effects data transfers directly between the memory and the slave processor over the memory data bus...Thus, data flows directly between the memory and the slave processor without passing through the master processor. The <u>only additional pins required by the master processor are the pins for the control signals</u> to the slave processor."

<u>Kozlowski et al</u>

Kozlowski et al is in a totally different processing field from Platko and concerns, col. 1, lines 14-29, processing of "image checks, deposit slips and other types of bank documents in which...document processing systems commonly contain multiple microprocessor elements that are responsible for different tasks within the machine. In addition to their individual processing responsibilities, these processor elements are required to communicate with each other. Latency occurs when a processor foregoes its own processing functions because it is waiting for a response from another processor. The lost time created by latency therefore significantly compromises the performance of multiple processor systems and has generally resulted in increased complexity in an effort to address the problem."

Kozlowski addresses this problem, col. 2, lines 31-59, by using, "a master processor 10 and a slave processor 20. <u>A multi-channel interface 30</u> is disposed between the slave processor 20 and the master processor 10. The multi-channel interface 30 has a low latency channel 32 for transferring low-latency information, and <u>a high-throughput channel 31</u> for transferring high-throughput information. The use of multiple channels thus <u>allows simultaneous transfers</u> of different data types...Thus, if the data to be transferred comprises high-throughput information, the transfer will take place across the high-throughput channel 31. Alternatively, if the data to be transferred comprises low-latency information, the transfer will take place across the low-latency channel 32."

Accordingly, Platko and Kozlowski are implemented in completely different processing fields and systems and have completely different objectives, which basically makes the systems noncombinable.

Platko concerns a network interface card as used in a PC, does not want a separate interface, and wants to reduce the number of pins consumed on an ASIC processor.

Kozlowski, on the other hand, uses a multi-channel interface having a low latency channel 32, a high throughput channel 31, and additional protocol flag interface channels 33.

In summary, the Platko system and the Kozlowski system are in totally different fields, address totally different problems, have totally different objectives and implement totally different processing systems, such that one skilled in the art would have absolutely no incentive or reason to attempt to combine the systems, as is attempted in the

Final Rejection in a very apparent and blatant attempt to combine the references through hindsight.

In fact, neither Platko nor Kozlowski is even close to addressing the technology of the present invention.

The arguments submitted above address the fact that the master processor of the present invention is substantially different from the master processor of Platko. Those arguments are not directed towards distinguishing the master processor as recited in the claims from the master processor of Platko. Rather, those arguments are directed towards the fact that the master processor of the present invention has a need and requirement for the data processing system of claims 1-13 and the method of claim 4. Whereas the master processor of Platko has no real need or requirement for the data processing system and method of the claims. Stated differently, there is absolutely no motivation or teaching leading one skilled in the art to modify Platko in view of the disclosure of Kozlowski to derive the subject matter of the present invention, particularly when considering the different technical fields and problems addressed by Platko and Kozlowski (and the present invention).

9. CONCLUSION

In view of the above, it is respectfully submitted that the Final Rejection is in error and should be reversed for good reasons, and it is respectfully requested that the Board of Patent Appeals and Interferences so find.

The required fee in the amount of $320.00 is herewith enclosed. The Commissioner is hereby authorized to charge any additional fees or credit any overpayment in connection herewith to our Deposit Account No. 19-1013/SSMP. A triplicate copy of this sheet is enclosed.

Respectfully submitted,

William C. Roch

William C. Roch
Registration No. 24,972

SCULLY, SCOTT, MURPHY & PRESSER
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343

WCR/jf

APPENDIX A

1. A data processing system allowing a master processor to access a memory by using a slave processor to access the memory while the master processor continues other processing tasks while the slave memory accesses the memory and writes the data from memory to the master processor comprising: a master processor; a slave processor; a memory; and a bus subsystem interconnecting the master processor, the slave processor, and the memory; wherein the master processor is configured to generate, in response to a memory access instruction, a read request comprising a read command for execution by the slave processor to read data stored in a location in the memory specified by the memory access instruction while the master processor continues other processing tasks while the slave processor accesses the memory and writes the data from memory to the master processor, and to write the read request to the slave processor via the bus subsystem, and the slave processor is configured to execute the read command received in the read request from the master processor to obtain the data stored at the specified location in the memory and to write the data obtained to the master processor via the bus subsystem while the master processor continues other processing tasks.

2. A data processing system as claimed in claim 1, wherein the bus subsystem comprises two buses interconnected by a bridge device and the slave processor is integrated in the bridge device.

3. A disk controller comprising: a master processor; a slave processor; a memory; and a bus subsystem interconnecting the master processor, the slave processor, and the memory; wherein the master processor is configured to generate, in response to a memory access instruction, a read request comprising a read command for execution by the slave processor to read data stored in a location in the memory specified by the memory access instruction while the master processor continues other processing tasks while the slave processor accesses the memory and writes the data from memory to the master processor, and to write the read request to the slave processor via the bus subsystem, and the slave processor is configured to execute the read command received in the read request from the master processor to obtain the

data stored at the specified location in the memory and to write the data obtained to the master processor via the bus subsystem while the master processor continues other processing tasks.

4. A method for reading data from a memory in a data processing system comprising a master processor, a slave processor, a memory, and a bus subsystem interconnecting the master processor, the slave processor, and the memory, the data processing system allowing a master processor to access a memory by using a slave processor to access the memory while the master processor continues other processing tasks while the slave memory accesses the memory and writes the data from memory to the master processor; the method comprising:

generating, by the master processor, in response to a memory access instruction, a read request comprising a read command for execution by the slave processor to read data stored in a location in the memory specified by the memory access instruction while the master processor continues other processing tasks while the slave processor accesses the memory and writes the data from memory to the master processor;

writing, by the master processor, the read request to the slave processor via the bus subsystem;

executing, by the slave processor, the read command received in the read request from the master processor to obtain the data stored at the specified location in the memory while the master processor continues other processing tasks; and,

writing the data obtained to the master processor via the bus subsystem.

APPENDIX B

1. Claims 1-4 have been amended and claims 1-4 are presented for examination.

2. The text of those sections of Title 35, U.S. Code not included in this office action can be found in a prior action.

3. Claims 1-4 are rejected under 35 U.S.C. 103(a) as being unpatentable over Platko et al. (US 6,330,658 B1), hereinafter Platko, in view of Kozlowski et al. (US 6,513,070 B1), Hereinafter Kozlowski.

4. As to claims 1 and 4, Platko discloses the invention substantially as claimed, including a data processing system (col. 1, lines 22-24) comprising:

a master processor (16, fig. 1; col. 2, lines 18-19);

a slave processor (18, fig. 1; col. 2, line 25);

a memory (20, fig. 1; col. 2, lines 19-20); and

a bus subsystem interconnecting the master processor, the slave processor, and the memory (fig. 1; col. 2, lines 18-30; col. 3, lines 20-25);

wherein the master processor is configured to generate, in response to a memory access instruction (i.e., read, write; col. 4, line 64-col. 5, line 13), a read request (col. 2, lines 45-49; col. 5, lines 35-47) comprising a read command for execution by the slave processor to read data stored in a location in the memory specified by the memory access instruction (col. 2, lines 32-35), and to write the read request to the slave processor via the bus subsystem (col. 2, lines 35-37), and the slave processor is configured to execute the read command received in the read request from the master processor to obtain the data stored at the specified location in the memory (col. 6, lines 3-24) and to write the data obtained to the master processor via the bus subsystem (col. 5, lines 35-44; col. 6, lines 3-24).

5. Platko does not specifically disclose allowing a master processor to continue other processing operations and tasks while slave processor is reading the memory. However, Kozlowski discloses allowing a master processor to continue other processing operations and tasks while slave processor is reading the memory (col. 1, lines 45-50 and 64-67; col. 2, lines 1-8). It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teaching of Platko and Kozlowski because Kozlowski's DMA operation would reduce latency by allowing the master processor to perform other tasks while data is being reading/writing (col. 1, lines 45-50 and 64-67; col. 2, lines 1-8).

6. As to claim 2, Platko discloses wherein the bus system comprises two buses interconnected by a bridge device (col. 2, lines 26-27).

7. As to claim 3, Platko discloses the invention substantially as claimed in claims 1 and 4. In addition, Platko discloses a disk controller (46, fig. 1; col. 5, lines 49-52).

8. Applicant's arguments with respect to claims 1-4 have been considered but are moot in view of the new ground(s) of rejection.

9. In the remarks, applicants argued in substance that

(1) Platko reference does not allow the master processor to continue other processing operations and tasks while the slave processor is reading the memory.

(2) The "master processor" in the present invention is a high performance processor which is entirely consumed while waiting for the data transfer, and is therefore expensive.

(3) In the cited prior art reference Platko, the master processor is synchronously sequencing the operation of the slave processor. Whereas, in the present invention, the master processor is sending an asynchronous request to the slave processor and the slave processor is coming back with an asynchronous replay.

10. Examiner respectfully traverses applicants' remark.

As to point (1), please see paragraph 5 above.

As to points (2) and (3), In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., asynchronous, high performance processor) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

11. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP §706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.13(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jungwon Chang whose telephone number is (703) 305-9669. The examiner can normally be reached on 8:30-6:00 (Monday-Friday).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (703) 308-9052. The fax phone numbers for the organization where this application or proceeding is assigned are (703) 746-7239 for regular communications and (703) 746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-9669.